Grammars on the hexagonal array

AUTHOR(S):

Aizawa, Kunio

CITATION:

Aizawa, Kunio. Grammars on the hexagonal array. 数理解析研究所講究録 1989, 695: 1-10

ISSUE DATE:

1989-06

URL:

http://hdl.handle.net/2433/101411

数理解析研究所講究録
第 695 巻 1989 年 1-10

Grammars on the hexagonal array

会沢 邦夫
(Kunio Aizawa)

Department Applied Mathematics, Faculty of Engineering, Hiroshima University
Higashi-Hiroshima, 724, JAPAN

Abstract

In this paper, we will define array grammars on the hexagonal grid. One of the advantages obtained from using the hexagonal grid is that each point has only one kind of neighbor. It is shown that the class of the context-free array grammars on the hexagonal grid includes the class of the rectangular ones even if their languages are restricted to the pictures connected on the rectangular grid. It is also shown that, in the monotonic case, they own same class of languages jointly.

1. Introduction

Array grammars were defined as the two-dimensional extensions of usual string grammars. The languages of these array grammars are embedded to the rectangular grid. In the processes to generate these languages, there is a shearing effect as the rows or columns of the host array must be stretched or shrunk by varying amounts in order to accommodate the array rewriting rules. To avoid the problem, a restriction called "isometric" is usually inserted to each production rule. Both-hand sides of an "isometric" production rule are geometrically identical. So the growth of an array can only happen along the border of the array by replacing the blank symbols (i.e., #s) with other symbols.

In this paper, we will define array grammars on the hexagonal grid. Since the definitions of the isometric array grammars are independent from the structures of arrays in

which they are embedded, the isometric grammars on the hexagonal array can be defined almost same as in the case of the rectangular grid. The hexagonal grid was defined formally in Mylopoulos and Pavlidis [1] and Rosenfeld [2] by making use of Abelian groups. For example, in [1], the rectangular grid is defined as a free Abelian group with two generators "go left", "go above", and their inverses "go right", "go below". The hexagonal grid is defined as an Abelian group with three generators and their inverses. One of the advantages obtained from using the hexagonal grid is that each point has only one kind of neighbor. This property is very useful to study topological properties of digital pictures. So introducing array grammars on the hexagonal array may shed light on the relationships between array grammars and digital pictures.

In Section 3 of this paper, we will compare the generative powers of the grammars on the hexagonal grid and the rectangular grid. It is shown that the class of the context-free array grammars on the hexagonal grid includes the class of the rectangular ones even if their languages are restricted to the pictures connected on the rectangular grid. It is also shown that, in the monotonic case, they own same class of languages jointly.

We assume the reader to be familiar with the theory of two-dimensional languages (otherwise please see, e.g., [3]).

2. Definitions and notations

We here will review some definitions and notations of the hexagonal grid. Then we will define two-dimensional grammars of the hexagonal array. At first, we review the definitions of discrete space and hexagonal array. We will use almost the same definitions and notations in Mylopoulos and Pavlidis [1].

Definition 1. A *discrete space* is a finitely presented abelian group $\Gamma=(X/D)$, where X has 2n generators $s_1, s_2, ..., s_n, s_1^{-1}, s_2^{-1}, ..., s_n^{-1}$, and D contains all relations other than the commutativity ($s_i s_j s_i^{-1} s_j^{-1}=1$) and the inverse iterations ($s_i s_i^{-1}=1$).

Definition 2. The *hexagonal array* is a discrete space described by a group Γ generated by the following relative directions: s_1= (right), s_2= (above right), s_3= (above left), s_1^{-1}= (left), s_2^{-1}= (below left), s_3^{-1}= (below right), and $D=\{s_1 s_3 s_2^{-1}=1\}$.

Note that Γ defined above is not a free group because (right)(above left)(below left) returns one to the original point. Hence D contains the relation $s_1 s_3 s_2^{-1}=1$. See Fig. 1 for a graphical image of the hexagonal grid and its generators.

Now, we review the definitions of some basic notions of discrete space. Let $X'=X\cup\{s_0\}$, where s_0 is the group identity.

Definition 3. For a given point x of $\Gamma=(X/D)$, a *direct neighbor* of x is any point $y\in\Gamma$ such that $y=xg$ for some $g\in X'$. The set of all direct neighbors of x is denoted by ND(x), i.e., $ND(x)=\{xg \mid g\in X'\}$.

Note here that ND(x) contains x itself. The set of direct neighbors of x excluding x is denoted as ND*(x), i.e., $ND^*(x)=ND(x)-\{x\}$.

Definition 4. The *neighborhood* of a point x, denoted as N(x), consists of

(1) ND(x) and

(2) all points z such that there exist y, $y'\in ND^*(x)$ with the property that the shortest path from y to y' not passing through z passes through z.

Again $N^*(x)=N(x)-\{x\}$.

Definition 5. A *direct-path* (d-path) between two points w_1 and w_2 of Γ is an ordered sequence of points $w_1=x_1, x_2, \ldots x_{n-1}, x_n=w_2$ such that $x_{i+1}\in ND^*(x_i)$, for $1\le i\le n-1$. A *path* is an ordered sequence of points such that $x_{i+1}\in N^*(x_i)$, for $1\le i\le n-1$. The *length of a (d-)path* is defined as the number of points in the path. A *shortest (d-)path* between two points is a (d-)path with minimum length.

Definition 6. Two points w_1 and w_2 of a finite subset F of Γ are *(d-)connected in F* if and only if there exists a (d-)path between them in F.

The relation "connected" defined above is obviously equivalence relation. Thus it partitions Γ into equivalence classes.

Definition 7. The *(d-)components* of Γ are the equivalence classes of "(d-)connected" relation.

The isometric grammars on the hexagonal array can be defined almost same as in the case of the rectangular array since their definitions are independent from the structures of arrays in which they are embedded.

Definition 8. An isometric grammar on the hexagonal array, denoted as IAG_h, is a construct G=<V, T, P, S, #>, where V is a finite nonempty set of nonterminals, T is a finite nonempty set of terminals ($V \cap T = \varnothing$), S is an element of V (the starting symbol), # is a symbol not contained in $V \cup T$ (the blank symbol), and P is a finite nonempty set of productions each of which is of the form $x \rightarrow y$. These x and y are geometrically identical arrays over $V \cup T \cup \{\#\}$ and satisfy the following conditions:

(1) If the non-#s of x do not touch the border of x, then the non-#s of y must be connected (and nonempty).

(2) Otherwise,

  (a) every connected component of non-#s in y must contain the intersection of some component of non-#s in x with the border of x;

  (b) conversely, every such intersection must be contained in some component of non-#s in y.

For the case of the rectangular array, it is shown in [3] that if conditions (1) and (2) defined above hold, applying the rule $x \rightarrow y$ does not disconnect or eliminate the non-#s. For the case of the hexagonal array, the same property can be shown in similar way.

Definition 9 Let W and Z are arrays over $V \cup T \cup \{\#\}$. For a given IAG_h, the array W directly derives array Z in G, denoted as $W \Rightarrow_G Z$, if there exists a production $x \rightarrow y$ in P, W contains x as a subarray, and Z is identical to W except that the subarray x is replaced with the array y. $\Rightarrow_G^*$ is defined as the reflexive and transitive closure of $\Rightarrow_G$. The *language* of G, denoted as L(G), is defined by $L(G)=\{W \mid W \text{ is an array over T and } S \Rightarrow_G^* W\}$.

An IAG_h is called *monotonic*, denoted as MAG_h, if #s are never created by any rule. We can also define *context-free* array grammar on the hexagonal array, denoted as $CFAG_h$, if G is an MAG_h and, for all rules $x \rightarrow y$ of G, x consists of a single nonterminal symbol and (possibly) of #s. Note here that all grammars defined above have 6-connectedness instead of 4-connectedness since each point of the hexagonal array has at most six direct neighbors.

3. Generative power of grammars on the hexagonal array

In this section, we will compare the generative power of grammars on the hexagonal and on the usual rectangular arrays. Because of the differences between the hexagonal and the rectangular arrays, it is difficult to compare these grammars directly. Thus we need an interpretative transformation between those two different arrays. Intuitively, if we regard the hexagonal array as a kind of brick array (see Fig 2) and slide each row to left (see Fig. 3), then we get a rectangular array. Namely the generator s_2=(above right) is regarded as s_2'=(above).

Stipulation 1. An array Σ on the hexagonal array, described by $\Gamma=(s_1, s_2, s_3, s_1^{-1}, s_2^{-1}, s_3^{-1}/s_1s_3s_2^{-1}=1)$, and an array Σ' on the discrete space $\Gamma'=(s_1, s_2', s_3, s_1^{-1}, s_2'^{-1}, s_3^{-1}/s_1s_3s_2'^{-1}=1)$ are said to be *equivalent under Stipulation 1*, denoted as $\Sigma=_1\Sigma'$, if $\Sigma=\Sigma'$ when the generators s_2=(above right) and s_2=(below left) of Γ are regarded as s_2'=(above) and s_2'=(below), respectively.

Note that the discrete space $\Gamma'=(s_1, s_2', s_3, s_1^{-1}, s_2'^{-1}, s_3^{-1}/s_1s_3s_2'^{-1}=1)$ defines slightly different array from usual rectangular array, i.e., the discrete space Ξ generated by s_1=(right), s_2=(above), s_1^{-1}=(left), $s_2'^{-1}$=(below), since each point of Γ' has six (not four) direct neighbors other than itself (see Fig. 4). Moreover it is different from the original hexagonal array since each point of Γ' has eight neighbors other than itself.

The class of IAG_h is obviously not equal to that of IAG even if we use the stipulation for there exist the languages of IAG_h whose elements are not d-connected under Ξ. In fact, even if we consider only the languages whose elements are d-connected under the discrete space Ξ, there exists $CFAG_h$ G such that L(G) is not contained in $\mathcal{F}$(CFAG).

Lemma 1. There exists a language of $CFAG_h$ which is d-connected under the discrete

space Ξ and is not contained in $\mathcal{F}$(CFAG).

Proof: Let G=<{S, A, B, C}, {a}, P, S, #> be a $CFAG_h$, where P consists the following production rules on the hexagonal array:

$$\begin{array}{l} {}^{\#}{}_{\#}{}^{\#}{}^{S} \\ {}^{\#}{}_{\#}{}^{\#} \rightarrow {}^{a}{}_{a}{}^{a}{}^{a}A \\ A\,{}^{\#}{}_{\#}{}^{\#} \rightarrow a\,a\,{}^{a}{}_{a}{}^{a}A \\ A\,{}^{\#}{}_{\#}{}^{\#} \rightarrow a\,a\,{}^{a}{}_{a}{}^{a}B \\ {}^{\#}{}_{B} \rightarrow {}^{C}{}_{a} \\ {}^{\#}{}_{\#}{}^{\#}\,C \rightarrow {}^{a}{}_{a}{}^{a}\,a\,a \\ \#\,\#\,C \rightarrow C\,a\,a \end{array}$$

An example of derivation steps of G is shown in Fig. 5. It is not so difficult to see that each element of L(G) is d-connected under the discrete space Ξ. Assume here that there exists a CFAG G' which generates the language of G. Since G' generates arrays embedded in the discrete space Ξ, each element of L(G') is of the form represented in Fig. 6. The top horizontal part of an element of L(G') must be generated independently from the rightmost vertical run (see Fig. 7). In general, the top horizontal part can be arbitrary long, then G' must use at least one nonterminal symbol more than once (see Fig. 8a). Thus, if G' generates the elements of L(G), G' also generates arrays which are not contained in L(G) (see Fig. 8b). It is a contradiction.

On the other hand, we will show that tape-bounded array acceptors accepts the class of languages of MAG_h which are d-connected under Ξ. A Turing array acceptor is a Turing machine defined on two-dimensional input tapes. If it "bounce off" #s, it is called tape-bounded (denoted as TBAA). It is known that the languages generated by MAGs are the same as the languages accepted by TBAA. For detail definitions and proofs, see, e.g., Rosenfeld [3].

Lemma 2. For any MAG_h G generating hexagonal arrays which are d-connected under the discrete space Ξ, there exists a tape-bounded Turing array acceptor A such that L(A)=L(G).

Proof: Given an MAG_h, the definition of a TBAA that accepts exactly L(G) is almost same

as that of TBAAs for the languages of MAGs. So, given an array Σ in the terminal alphabet T of G, embedded in an infinite array of #s, A moves around nondeterministically, starting at some point of Σ, and rewrites each non-# symbol p into (p, #). Finally, at some step, A rewrites some p as (p, S); this can happen only once. After it happens, A begins to simulate rules of G on the second terms of pairs. A problem arises with the TBAA if G has a rule $x \rightarrow y$ whose non-#s of right-hand sides are not d-connected under Ξ. These disconnected productions may raise disconnected parts on Σ (from the assumption, of course, Σ is d-connected as a whole). A must across the disconnected part in order to apply the rule. Suppose that A is located at position (i, j) and wants to across the position (i', j'), then the absolute values of the differences $|i-i'|$ and $|j-j'|$ are equal to 1 and both points are on the same border of Σ. To find the point (i', j'), A marks (i, j), follows the border of Σ, and keeps track of its net up, down, left, right moves. If A returns to (i, j) without finding the point (i', j'), the rule $x \rightarrow y$ cannot apply to the part. Otherwise A can continue its simulation.

It is obvious that the class of the languages of $CFAG_h$ (MAG_h) includes the class of the languages of CFAG (MAG, respectively) in the sense of Stipulation 1. So it is not so difficult to see the following theorem:

Theorem (1) The class of the languages of $CFAG_h$s which are d-connected under the discrete space Ξ includes the class of the languages of CFAGs properly in the sense of Stipulation 1.

(2) The class of the language of MAG_hs which are d-connected under the discrete space Ξ is equal to the class of the languages of MAGs in the sense of Stipulation 1.

5. Concluding Remarks

In this paper, we have shown the generative powers of some array grammars generating hexagonal patterns. The hexagonal array used in this paper is also defined as a near abelian partial path group of degree 6 (see [2]). Graph grammars using path controlled embedding, PCE grammars, can define the structure of their language by making use of partial path group. So it seems to be interesting to consider the PCE grammars which generate hexagonal array languages.

References

[1] Mylopoulos, J.P., and T. Pavlidis: On the topological properties of quantized spaces, *J. ACM*, 18, pp.239-254, 1971

[2] Rosenfeld, A.,: Partial path groups and parallel graph constructions, in G. Rozenberg and A. Salomaa (eds.), *The Book of L*, Springer-Verlag, pp.369-382, 1986.

[3] Rosenfeld, A.,: *Picture Languages*, Academic Press, 1979

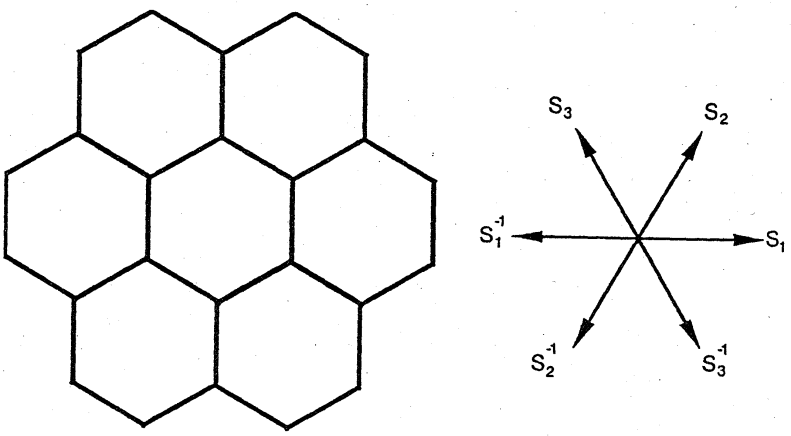


Figure 1. The hexagonal array and its generators.

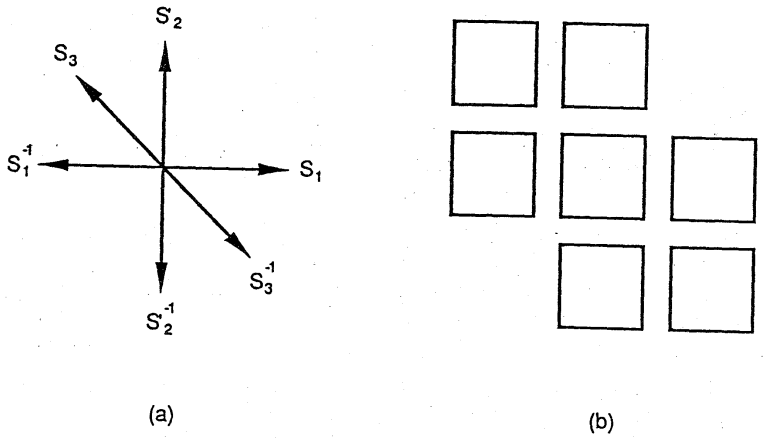


Figure 4. (a) Generators of Γ'; (b) the direct neighbors of an arbitrary point in Γ'.

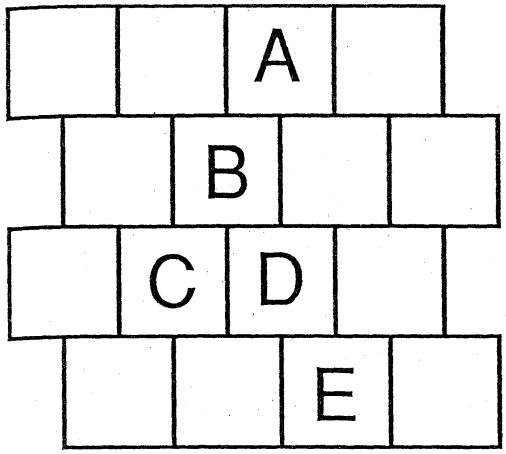


Figure 2. The blick array.

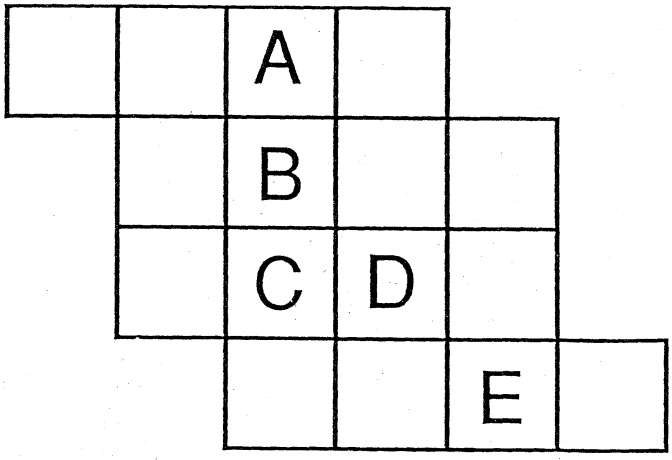


Figure 3. The rectangular array.

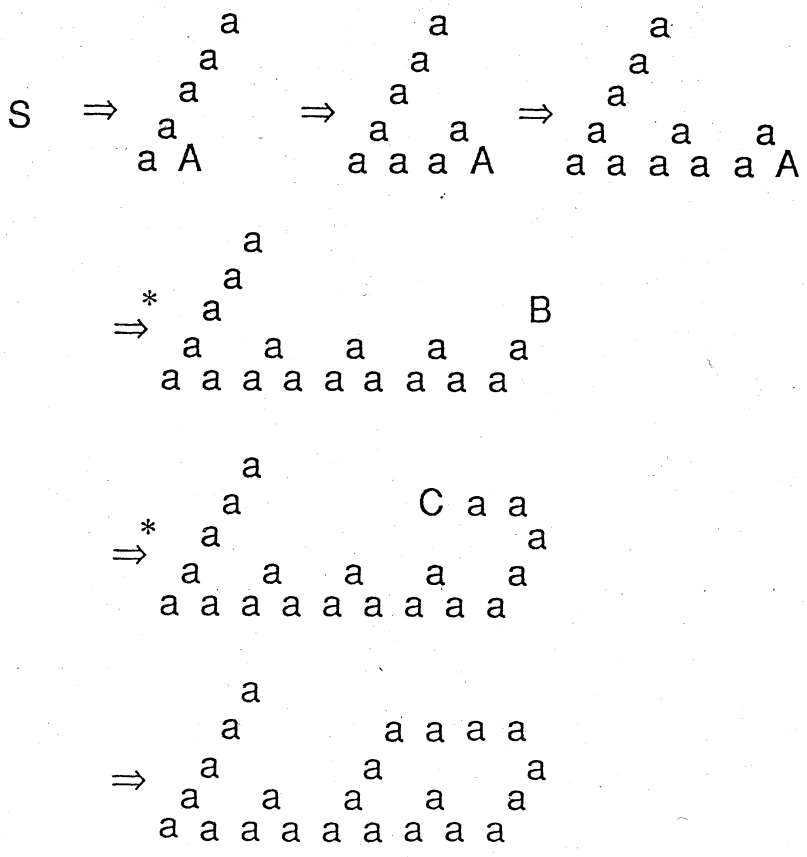


Figure 5. An example of derivations of G.

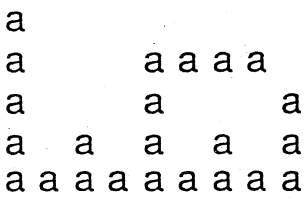


Figure 6. An example of elements of L(G').

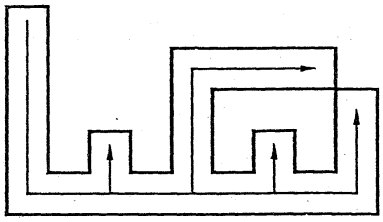
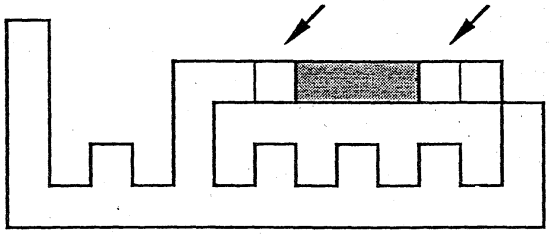
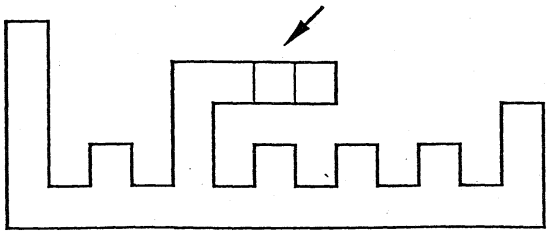


Figure 7. An example of generation of G'.



(a)



(b)

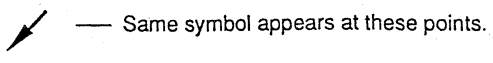


Figure 8. (a) An element of L(G') which is also in L(G), and
(b) an element of L(G') which is not in L(G).